CHALENCON Yoan

8

19,5/20

4 Valeur de retour d'un processus

Tout processus Unix retourne à son père une valeur correspondant au résultat de son exécution :

— 0 si tout s'est bien passé ;

— $\neq 0$ s'il y a eu une erreur.

Nous verrons dans un prochain TP comment utiliser cette valeur mais nous pouvons déjà l'observer en utilisant la variable `$?` du shell.

24. Lancez `ls` puis faite afficher `$?`. Quelle est sa valeur ?

ls
echo $? → la valeur est 0, ls s'est exécutée sans erreur

25. Même question avec `ls trucmachin`.

ls trucmachin
echo $? → la valeur est 2, ls a rencontré une erreur (il n'y a pas de dossier ou fichier trucmachin)

26. Récupérez le script shell `/users/but/info/Public/valeurRetour`. Ouvrez-le pour observer son contenu puis testez les 4 cas possibles et vérifiez en affichant `$?`.

cp /users/but/info/Public/valeurRetour
valeurRetour → echo $? → 99
valeurRetour un → echo $? → 1
valeurRetour deux → echo $? → 2
valeurRetour truc → echo $? → 0

Compte-rendu

Vous rendrez à l'enseignant un exemplaire du sujet complété à la fin du TP. Si vous n'avez pas fini, utilisez l'exemplaire qui vous reste pour terminer pendant votre temps libre.

Utilisation du shell Unix

5 - Processus

Rappel : vous devez simplement effectuer le travail demandé et noter les commandes utilisées et les réponses aux questions dans l'espace laissé libre. **Attention :** l'objectif est que vous soyez capables de refaire SEUL les manipulations décrites.

Pré-requis

Vous devez avoir parcouru les chapitres « Le langage de commande » et « Les processus » dans les diapos de cours : R1.04-IntroSysteme/Unix.pdf

1 Rôle du `PATH`

La variable `PATH` contient la liste des répertoires où le shell va chercher les commandes à exécuter.

1. Donnez la commande qui affiche la valeur de votre PATH.

echo $PATH

2. Copiez le fichier `/users/but/info/Public/bravo.c` dans votre répertoire de TP puis compilez-le et lancez-le (3 commandes).

cp /users/but/info/Public/bravo.c Cours/R1.04/TP/TP5 (cd Cours/R1.04/TP/TP5)
gcc -Wall bravo.c -o bravo
./bravo

3. Annulez la valeur de la variable `PATH`, puis lancez la commande `ls`. Expliquez ce qui se passe.

unset PATH
ls → on obtient une erreur disant que la commande n'existe pas ("Aucun fichier ou dossier de ce type")

4. Donnez la commande qui permet de lancer quand même ls.

/usr/bin/ls

5. Définissez `PATH` avec les deux répertoires `/bin` et `/usr/local/bin`. Vérifiez que vous arrivez à lancer `ls` en tapant simplement `'ls'`.

PATH=/bin:/usr/local/bin
ls → la commande fonctionne à nouveau

6. Lancez le programme `bravo`. Que se passe-t-il ? Expliquer.

bravo → on obtient une erreur disant que la commande est introuvable. Cela survient car bravo est dans le répertoire courant qui n'est pas dans le PATH.

7. Vous pouvez toujours lancer `bravo` en écrivant simplement `./bravo`, mais on préfère souvent ajouter le répertoire courant au `PATH`. Ajoutez le répertoire courant au `PATH` et vérifiez que vous arrivez à lancer `bravo` en tapant simplement `bravo`.

PATH=/bin:/usr/local/bin:.
bravo → la commande fonctionne.

Notez que vous retrouvez le `PATH` d'origine en ouvrant une nouvelle fenêtre terminal (raccourci CTRL-SHIFT-N).

2 Processus

Un processus Unix est toujours créé par un autre processus, appelé son *père*.

8. Lancez en **arrière-plan** (faire suivre la commande d'un `&`) un éditeur de texte (`gedit` ou `geany`).

gedit &

9. Affichez la liste des fils du processus shell courant avec la commande `ps -f`. Comment vérifie-t-on que `bash` est bien le père du processus que vous venez de lancer (`gedit` ou `geany`) ?

ps -f
le processus gedit a pour père le processus avec l'ID 31117 qui correspond à l'ID du processus bash. Donc bash est bien père du processus gedit.

10. Qui est le père de la commande `ps` elle-même ?

La commande ps a aussi pour père bash.

11. Donnez les commandes qui permettent de vérifier que la variable `$$` contient bien le numéro du processus shell courant.

echo $$ → 31117
ps -f → l'ID du processus bash est bien 31117

12. Affichez la liste de tous les processus avec `ps -ef`, observez-la en créant un tube sur `less`.

ps -ef | less

13. Affichez la liste de tous les processus avec `ps -ef` et filtrez cette liste avec `grep` pour ne retenir que les processus qui vous concernent.

ps -ef | grep chalency

14. Essayez la commande `ps -fU` *`votreLogin`* qui n'affiche que les processus de l'utilisateur indiqué (*votreLogin*). Filtrez le résultat avec `grep` pour ne retenir que les processus `bash`. Qui est le père des processus `bash` ?

ps -fU chalency | grep bash /ps -fU chalency|grep 30006
Le père des processus bash est le processus gnome-terminal-server

15. La commande `ps -ef --forest` affiche la liste de tous les processus sous forme d'arbre. Avec cette commande et `less`, déterminez le chemin complet depuis `bash` jusqu'à la racine de l'arbre (ne donnez que les noms des commandes, sans le répertoire ni les options).

ps -ef --forest
bash → gnome-terminal-server → systemd → init

16. Quel est le nom et le numéro du processus à la racine de l'arbre ?

Le processus à la racine est init avec l'ID 1.

3 Processus et signaux

La commande `kill -`*`signal num`*`...` envoie un *signal* aux processus dont le numéro est donné. Le signal le plus utilisé est `INT` qui interrompt le processus. Ainsi l'appui sur Ctrl-C pendant l'exécution d'un programme demande au shell d'envoyer un signal `INT` au processus en cours.

Récupérez le programme `/users/but/info/Public/boucle.c` et modifiez-le pour qu'il affiche indéfiniment des carreaux. Recompilez-le puis lancez son exécution et arrêtez-le avec Ctrl-C.

17. Lancez `boucle` en *arrière-plan* et tentez de l'arrêter avec Ctrl-C. Que se passe-t-il ? Expliquez.

./boucle &
On ne peut pas l'arrêter car Ctrl-C arrête le processus en cours mais nous avons ouvert boucle en arrière-plan

18. Ouvrez un nouveau terminal (si nécessaire) puis avec `ps` déterminez le numéro du processus `boucle` et le numéro de son père.

ps -ef → le processus a pour ID 35035 et son père a l'ID 31117

19. Envoyez un signal `INT` au processus `boucle` avec `kill`. Que se passe-t-il ?. Expliquez.

kill -INT 35035 → le processus boucle s'arrête, les carreaux ne s'affiche plus.

20. Envoyez maintenant un signal `INT` au père de `boucle` (le shell qui l'a lancé). Que se passe-t-il ?

kill -INT 31117 → Une nouvelle ligne apparait dans le terminal qui exécutait boucle, permettant de saisir de nouvelles commande.

Vous pouvez vérifiez l'effet d'un signal `INT` sur un shell en tapant Ctrl-C dans un terminal.

Note : le shell qui s'exécute dans une fenêtre `gnome-terminal` lit les commandes sur son entrée standard (le clavier par défaut). Si l'entrée se termine, le shell se termine et la commande se ferme. On peut donc fermer un terminal en tapant Ctrl-D (marque de fin de fichier). Essayez sur un de vos terminaux ouverts.

21. Envoyez maintenant le signal `KILL` au shell. Que se passe-t-il ?

ps -ef | grep gnome-terminal
kill -KILL 30006 → le terminal se ferme

Contrairement au signal `INT`, le signal `KILL` ne peut pas être intercepté par le processus et il force l'arrêt brutal du programme. À ce titre, il ne doit pas être utilisé pour arrêter un programme qui peut l'être par d'autres moyens. Ainsi, si vous *tuez* un processus `geany` ou `gvim`, les modifications qui n'ont pas été enregistrées par l'éditeur seront perdues.

22. On peut lancer un processus en arrière-plan en faisant suivre la commande du caractère `&`. La commande `xterm` ouvre un terminal simple. Donnez la commande lançant en une fois 4 processus `xterm`.

xterm & xterm & xterm & xterm &

23. Déterminez avec `ps` les numéros des 4 processus `xterm` lancés en arrière plan puis donnez la commande qui envoie un signal `INT` aux quatre processus.

ps -ef | grep xterm → 37238, 37239, 37240, 37241
kill -INT 37238 37239 37240 37241